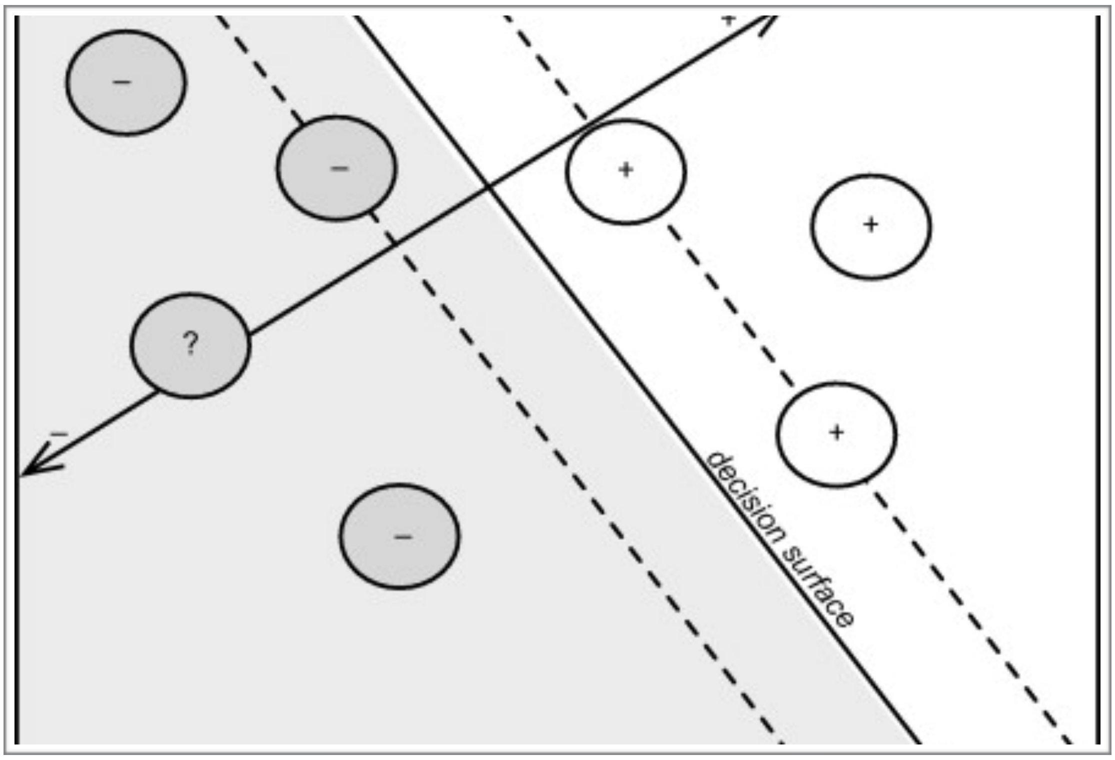
# COMP9417

*Machine Learning & Data Mining*



## Assignment 2

## Topic 0: Implementation of SVM on CCI Market Data

By

Elliot Griffiths

z3332776

## 1. Introduction

Efficient market hypothesis tells us that the market cannot be 'beaten' with all available information instantly priced into the fair value. However, the focus on automated trading systems has become more prominent in the last decade. Investment banks are opting to hire computer scientists over traders with companies such as Goldman Sachs replacing 600 of their traders with 200 software engineers. Clearly there is a reason for all the interest.

The goal of this project was to implement a set of features to identify overbought and oversold levels in the underlying security / stock and use these features to learn classifiers using a Support Vector Machine algorithm. The features chosen were a set of 29 Commodity Channel Index (CCI) oscillators ranging from 2 to 30 period calculation.

The Commodity Channel Index indicator is used to identify extremes in stock, ETF and indices trends. It does this by measuring the relative difference between an average price level over a set number of periods and the current price level. The exact formula is as follows;

CCI = (MBP - n-period Moving average of MBP) / (C x Mean Deviation)
where
Mean Bar Price (MBP) = (Bar High + Bar Low + Bar Open) / 3
Constant (C) = 0.015

# 2. Implementation

## 2.1 Language selection

R was chosen for several reasons;

- Collection - The Quantmod package in R allows convenient download and import of data from yahoo finance. It is also great for feature generation.
- Manipulation - The dplyr package allows relatively easy column selection, sorting and filtering.
- Visualisation - ggplot2 offers insight into data with minimal difficulty.

## 2.2 Data Collection & Manipulation

In todays world of algorithm trading the growing majority use extremely short time periods (high frequency trading). The use of relatively simplistic models working on tick by tick data has shown some of the greatest promise. Unfortunately access to this data is expensive. Thus I had to settle for free daily data sources via yahoo finance.

Daily stock data was collected from yahoo finance using R's Quantmod package and getSymbols() function.
getSymbols also conveniently warns of missing values in the data, such as the example below of the S&P500 index data.

```
> getSymbols('^GSPC', from='2015-01-01')
[1] "GSPC"
Warning message:
^GSPC contains missing values. Some functions will not work if objects contain
missing values in the middle of the series.
```

Missing data was removed using na.omit() function. High, low and close prices where extracted from the raw data using HLC() function. Log daily returns were then calculated using dailyReturn( <SYMBOL>, type='log') function.

## 2.3 Target & Feature Generation

Firstly the targets were generated using the sign of the log returns. 1 was set for positive return days and -1 for losing days. The features where then generated using quantmod's built in indicator functions (e.g CCI, SMI, ADX, etc ).
After the indicators where calculated they were then lagged to avoid look-ahead bias. This was to ensure the features used for clarification did not include the data from the day in which we were trying to predict. i.e. the data seen by the algorithm to predict the current day only consisted of data up to the day prior.

## 2.4 Algorithm Selection

Algorithm Requirements:

1. Able to model nonlinear/complex relationships.
2. Fast to predict new instances. (For running on minute / tick bar charts in future testing)
3. Able to handle noise from varying market conditions
4. Less prone to overfitting

Algorithms considered:

| **Algorithm** | **Sentiment (-/5)** | **Reason** |
|---|---|---|
| Linear Regression | 1/5 | Not able to fit complex relationship |
| Neural Network | 3/5 | Long training time. More prone to overfitting |
| K-Nearest Neighbours | 2/5 | Slow to predict new instances (bad for high frequency trading) |
| Support Vector Machine | 4/5 | Resistant to noice with max margins around decision boundary. Can handle complex relationships |

Thus SVM was chosen

### 2.5 Algorithm Implementation

The SVM was implemented using R's ksvm() function. A Gaussian kernel function was used due to the complex non-linear features. (Parameter selection will be discussed later in section 3.) Once the model was trained we then made predictions from the data at time period t+1 while excluding the targets in column 1. We use period t+1 as we want to look at the current data to predict the decision for tomorrows movements. The predictions are then multiplied by the market returns to produce the returns of the strategy.

## 3. Experimentation

### 3.1 Features

Multiple combinations of features were tested including ADX, CCI, SMI. Each showed varying results. Further analysis and optimisation of combinations of features would be have been undertaken if I was working in a team and not alone.

### 3.2 Learning Algorithm Parameters

3.2.1 Crossvalidation
10 - fold cross-validation was set to ensure overfitting was avoided.

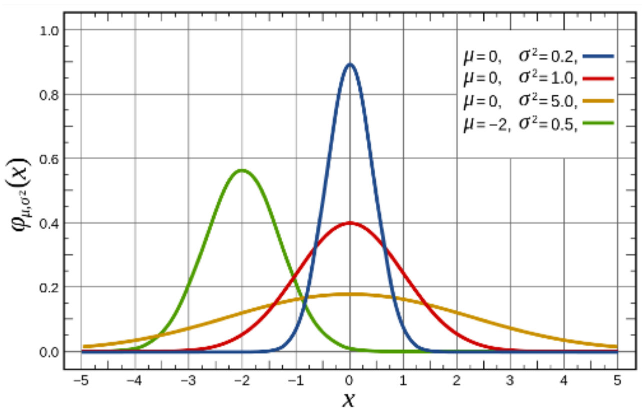
3.2.2 C

The C parameter is used to decide how much 'weight' to give the incorrectly classified points. A high C will cause the margins between the classified classes to decrease as the decision boundary will weigh the incorrect points higher. Experimentation with C values showed C = 60 to be most optimal. This was due to the data having not a clear separation as can be seen in figures 4.1.2.

### 3.2.3 Sigma

The Sigma parameter controls the width of the gaussian kernel function. A higher value will cause the decision boundaries to have less variation. Testing on sigmas from 0.05 - 5 was performed. Variance increased with lower sigma values as the model fitted the training data to a higher degree 'less flat'. In our case 0.05 sigma was optimal. See 4.1.2.

Sigma affect on normal distribution:



Source : Available at: https://thecuriousastronomer.wordpress.com/2014/06/26/what-does-a-1-sigma-3-sigma-or-5-sigma-detection-mean/. [Accessed 04 June 2017].

## 3.3 Data Set

Interestingly the algorithm was able to learn specific stocks with better returns than others. For example the model beat the bench marks by approximately 20% for AAPL and AMAZ. Shown in 4.1.1. Where as TSLA was about equal to the benchmark buy and hold strategy.

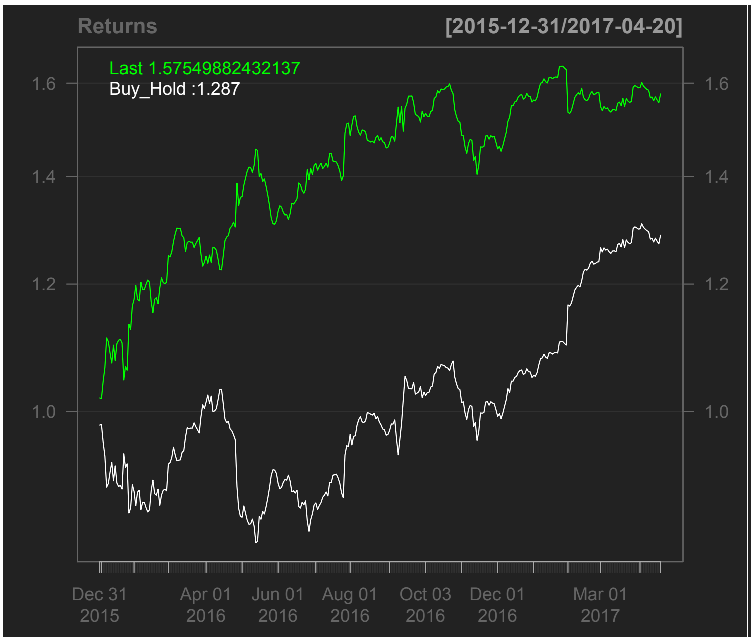
# 4. Results

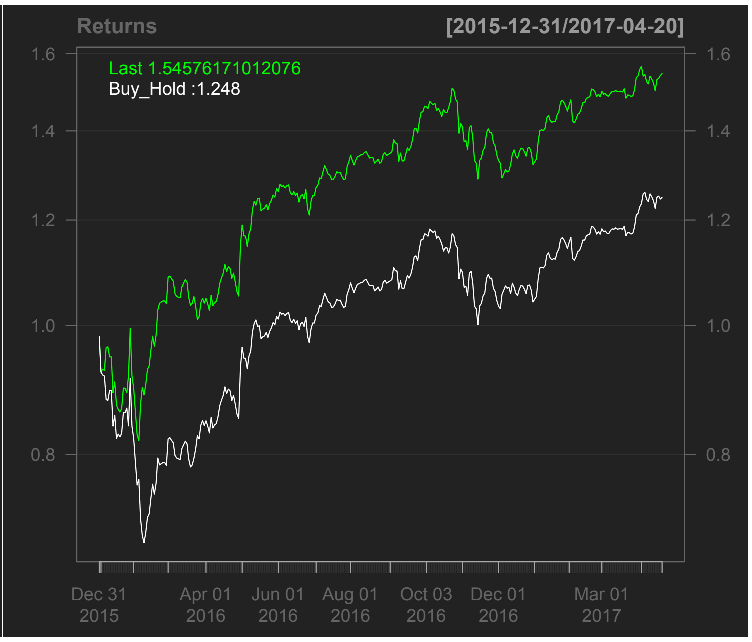
## 4.1.1

Last indicates the return of the strategy over the period while Buy_Hold indicates the return of the underlying security.

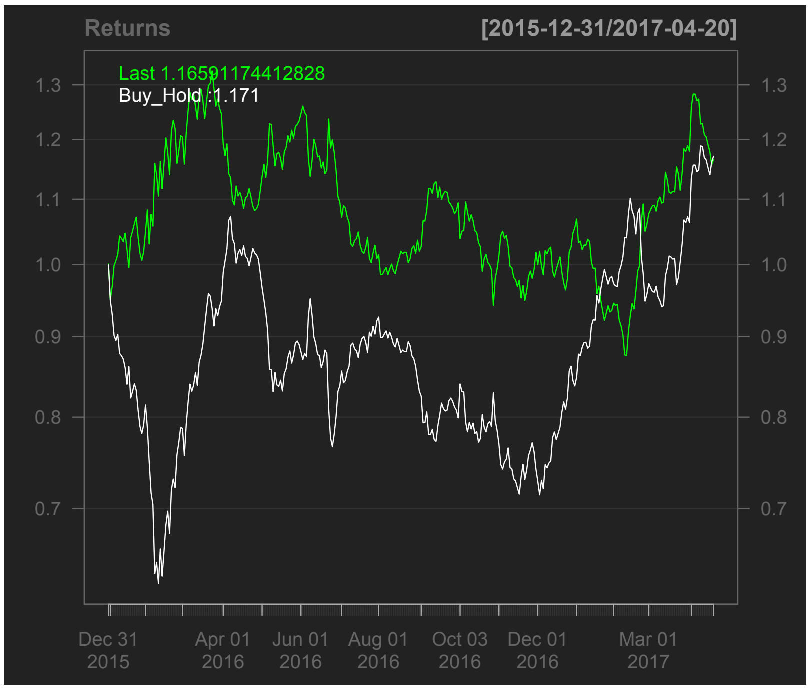
AAPL

AMZN

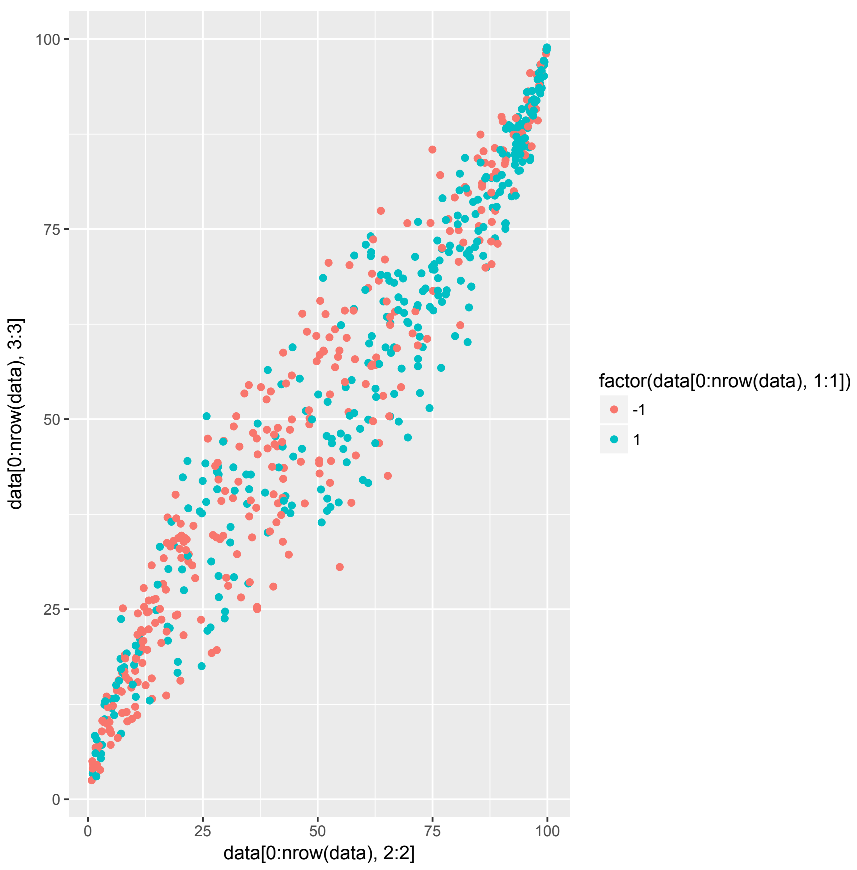




TSLA

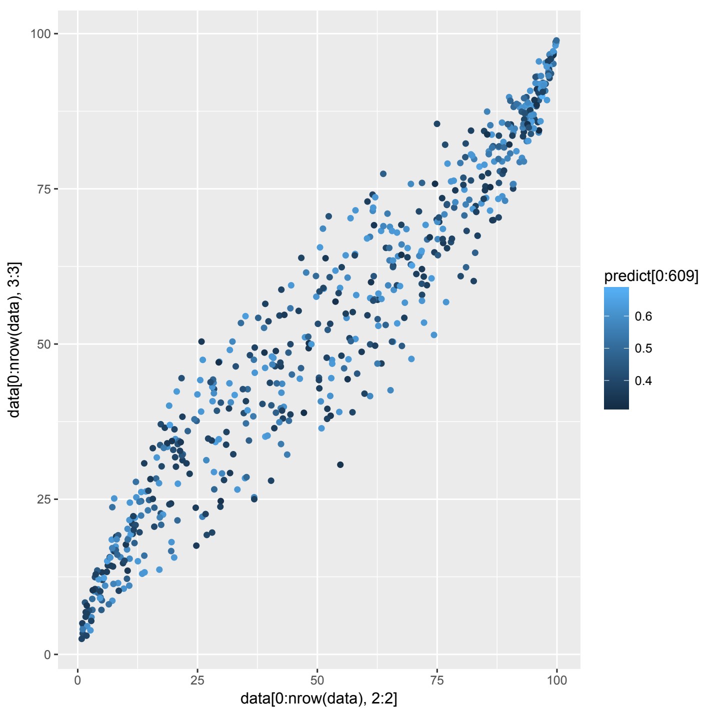


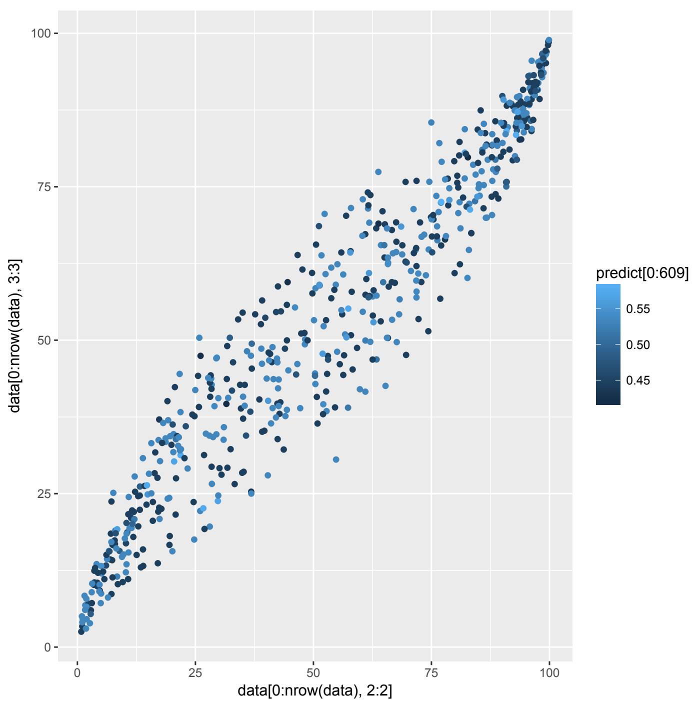
### 4.1.2



Sigma = 0.05
Correct = 54.25%

Sigma = 5
Correct 51.51





## Conclusion

While the model showed promising result in some tests, the over 55% correct prediction was limited for specific data sets at various times. Further research in feature selection needs to be undertaken to increase the accuracy of the model. Alternative learning algorithms may also be beneficial.

## References

- ksvm function | R Documentation. 2017. *ksvm function | R Documentation*. [ONLINE] Available at: https://www.rdocumentation.org/packages/kernlab/versions/0.9-25/topics/ksvm. [Accessed 01 June 2017].

- M., T, 1997. *Machine Learning*. 1. McGraw-Hill Education
- Thecuriousastronomer. 2017. *What does a 1-sigma, a 3-sigma or a 5-sigma detection mean? | thecuriousastronomer*. [ONLINE] Available at: https://thecuriousastronomer.wordpress.com/2014/06/26/what-does-a-1-sigma-3-sigma-or-5-sigma-detection-mean/. [Accessed 04 June 2017].